

本节内容

分块查找

知识总览



算法思想

分块查找



查找效率分析 (ASL)

分块查找的算法思想

“索引表”中保存每个分块的最大关键字和分块的存储区间

10	20	30	40	50
[0,1]	[2,5]	[6,8]	[9,10]	[11,13]

0 1 2 3 4

7	10	13	19	16	20	27	22	30	40	36	43	50	48	...
0	1	2	3	4	5	6	7	8	9	10	11	12	13	...

//索引表

```
typedef struct {  
    ElemType maxVal;  
    int low,high;  
}Index;
```

//顺序表存储实际元素

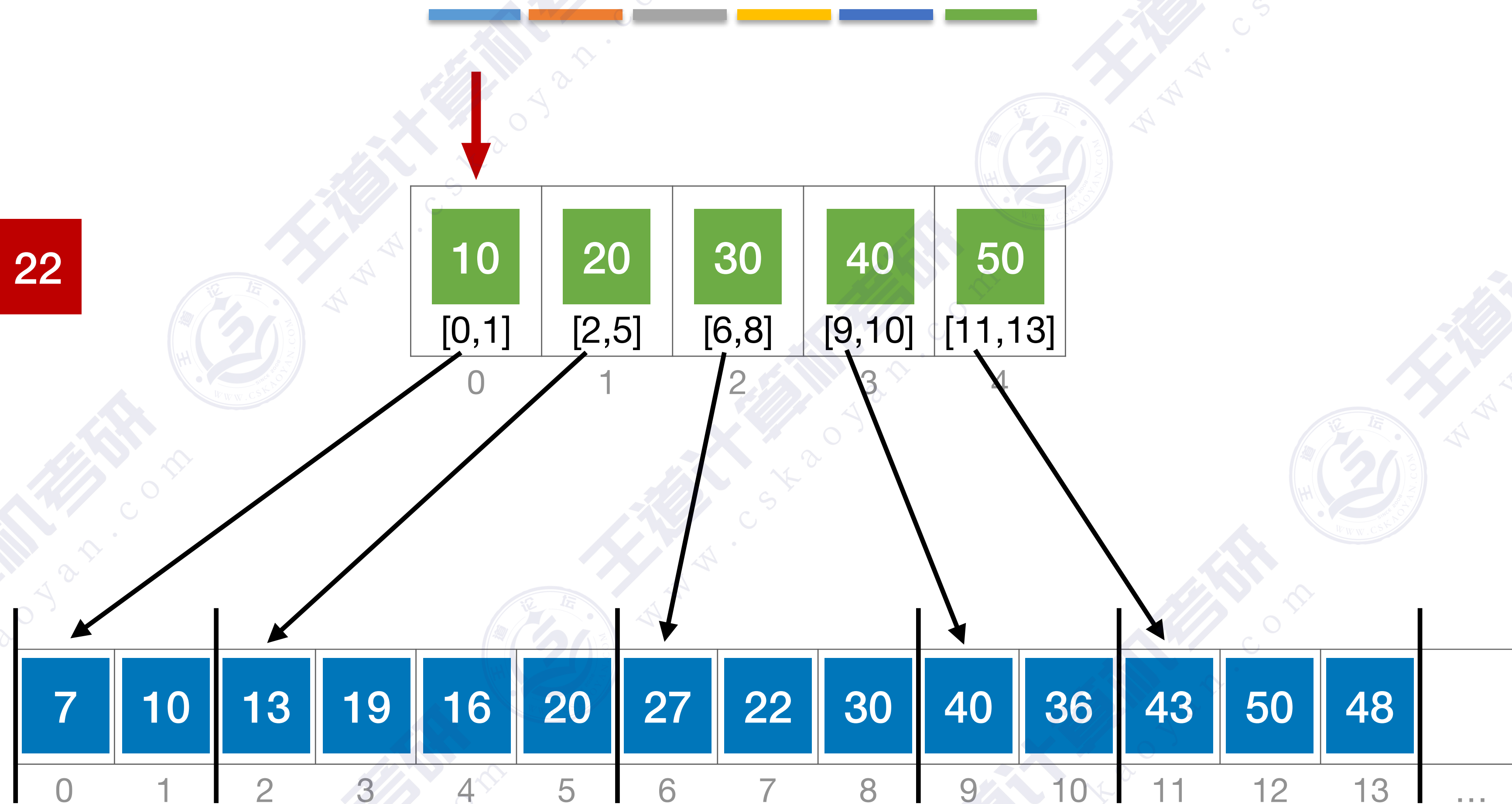
```
ElemType List[100];
```

特点：块内无序、块间有序

分块查找的算法思想

查找目标:

22



分块查找的算法思想

查找目标:

22

10	20	30	40	50
[0,1]	[2,5]	[6,8]	[9,10]	[11,13]

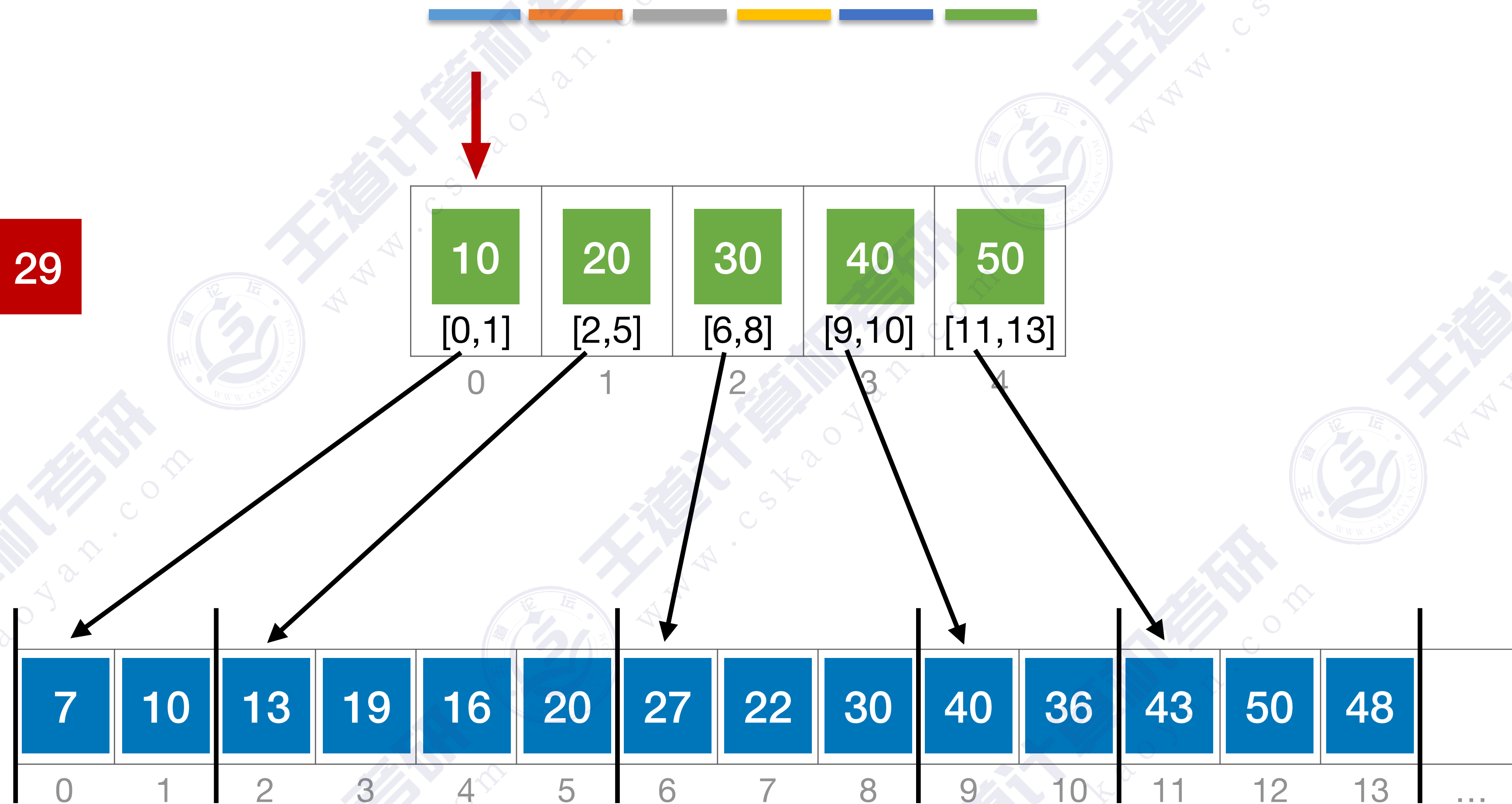
7	10	13	19	16	20	27	22	30	40	36	43	50	48	...
0	1	2	3	4	5	6	7	8	9	10	11	12	13	...

查找成功

分块查找的算法思想

查找目标:

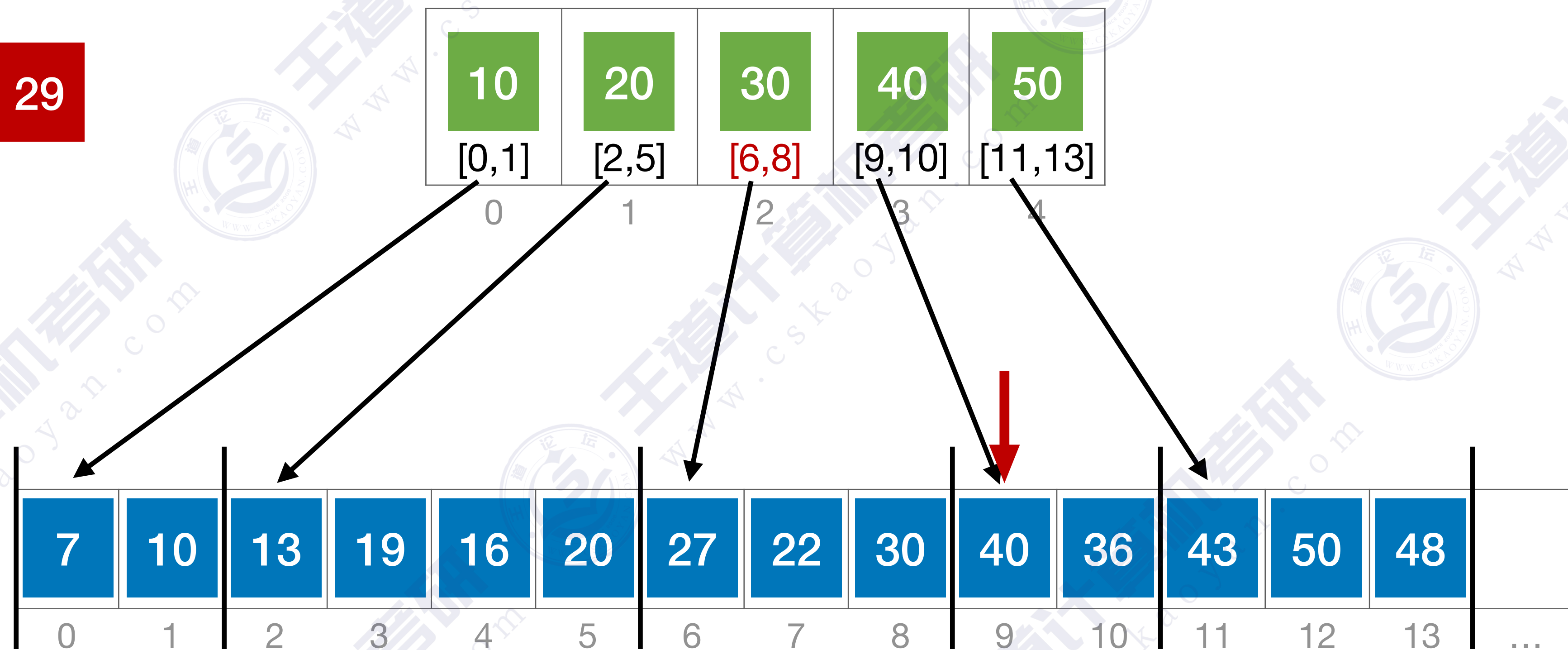
29



分块查找的算法思想

查找目标:

29



超出分块范围，查找失败

分块查找的算法思想

“索引表”中保存每个分块的最大关键字和分块的存储区间

10	20	30	40	50
[0,1]	[2,5]	[6,8]	[9,10]	[11,13]

0 1 2 3 4

7	10	13	19	16	20	27	22	30	40	36	43	50	48	...
0	1	2	3	4	5	6	7	8	9	10	11	12	13	...

//索引表

```
typedef struct {  
    ElemType maxVal;  
    int low,high;  
}Index;
```

//顺序表存储实际元素

```
ElemType List[100];
```

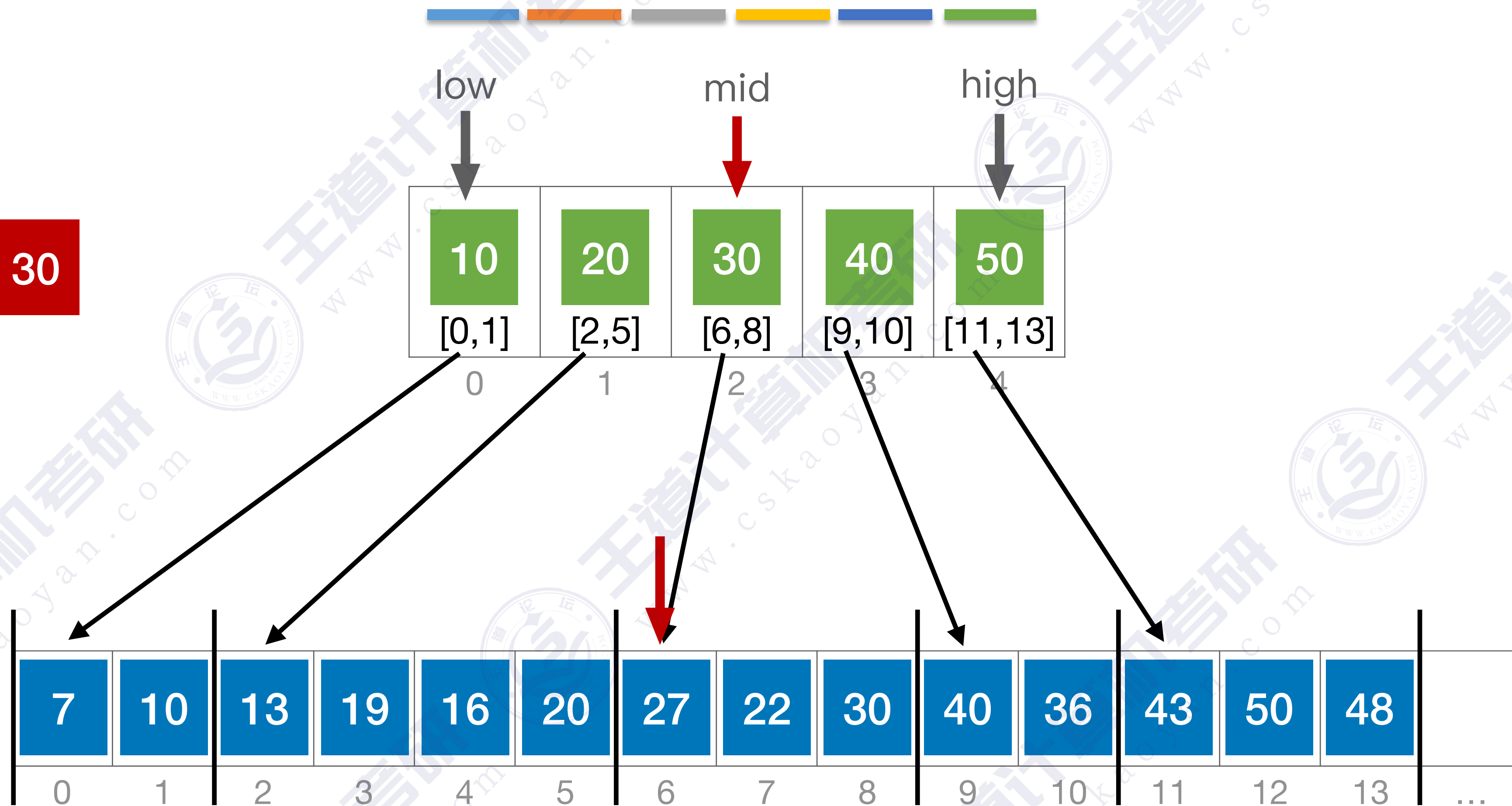
分块查找，又称索引顺序查找，算法过程如下：

- ①在索引表中确定待查记录所属的分块（可顺序、可折半）
- ②在块内顺序查找

用折半查找索引

查找目标:

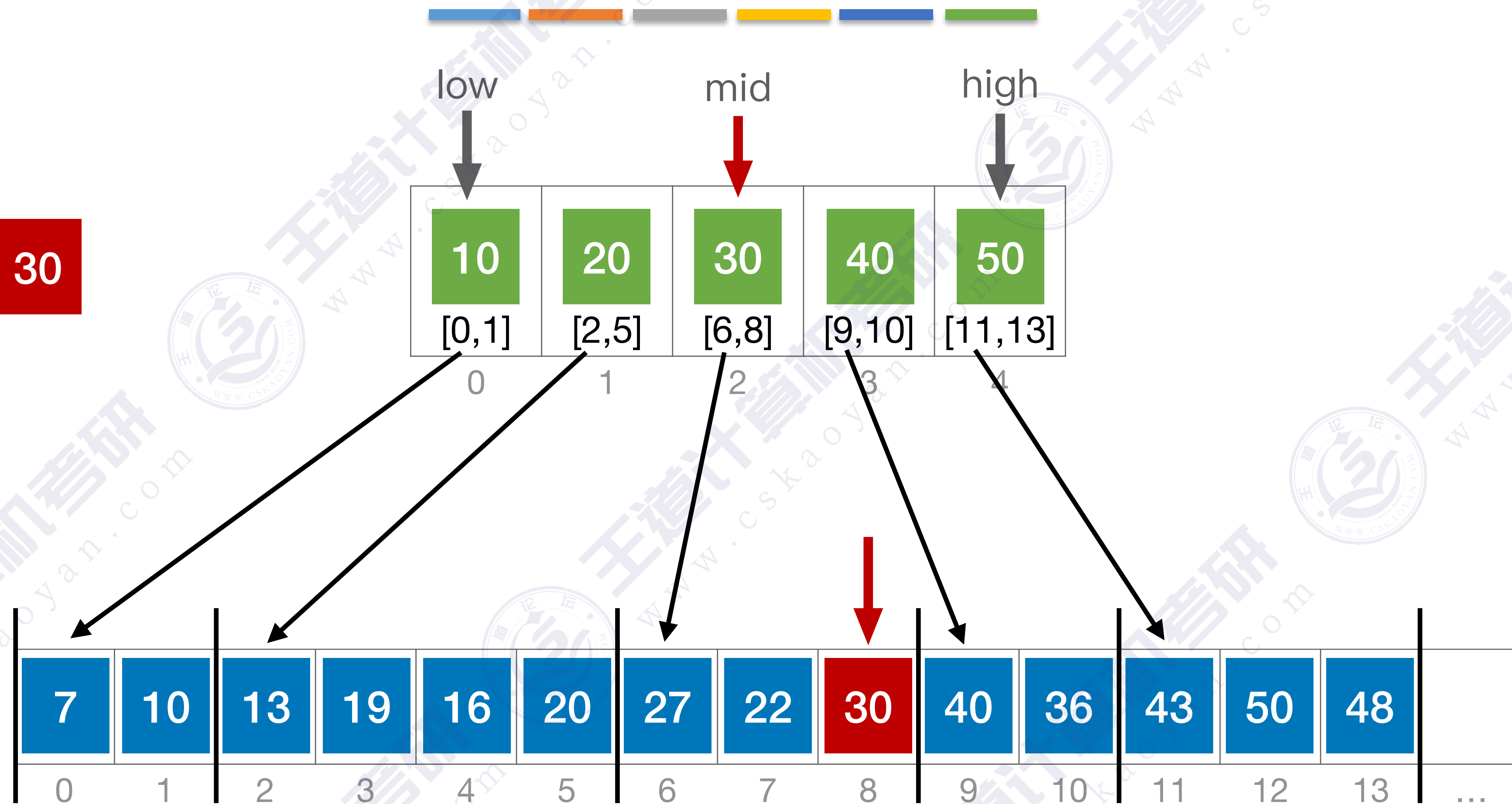
30



用折半查找索引

查找目标:

30

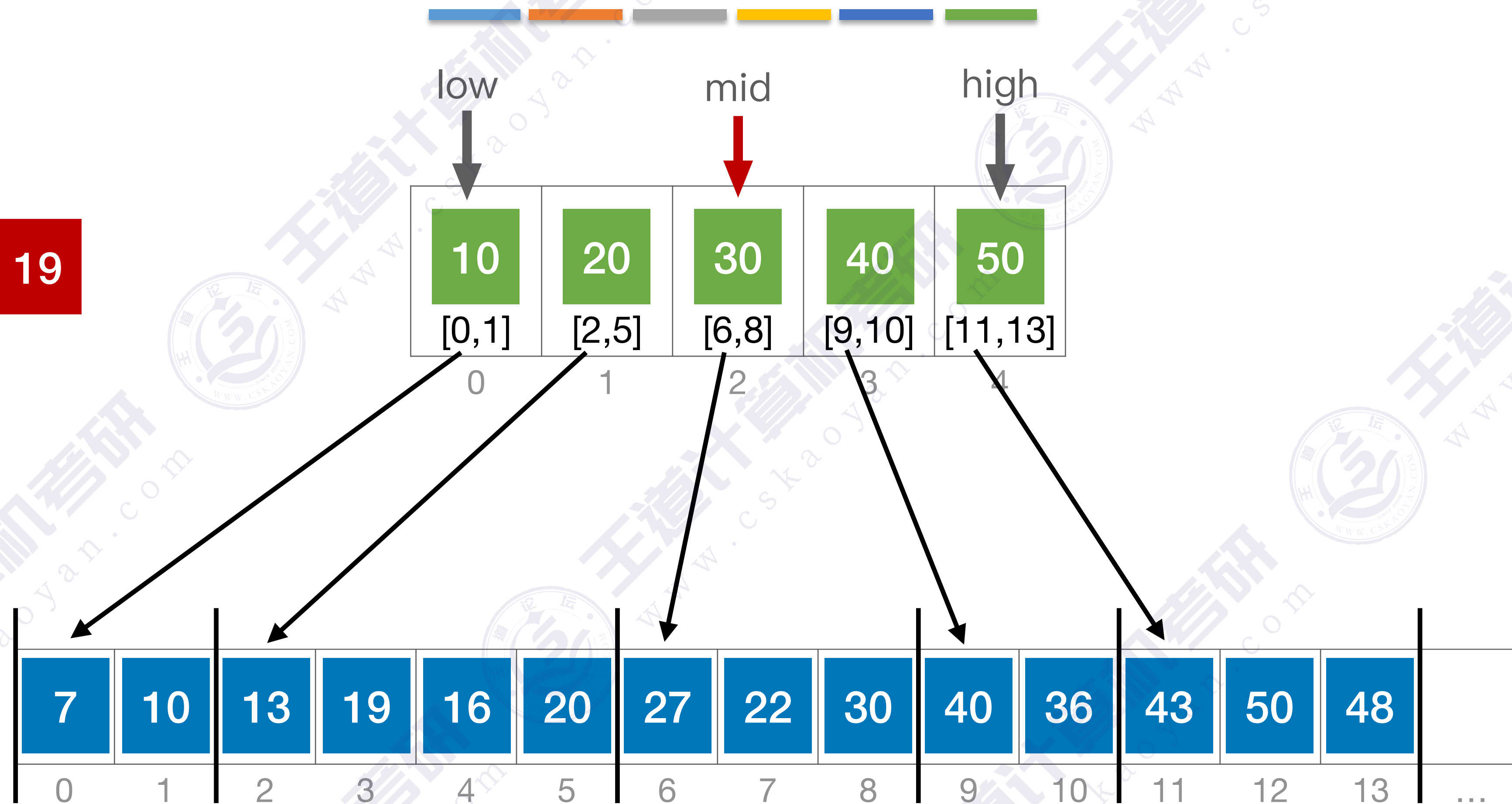


查找成功

用折半查找查索引

查找目标:

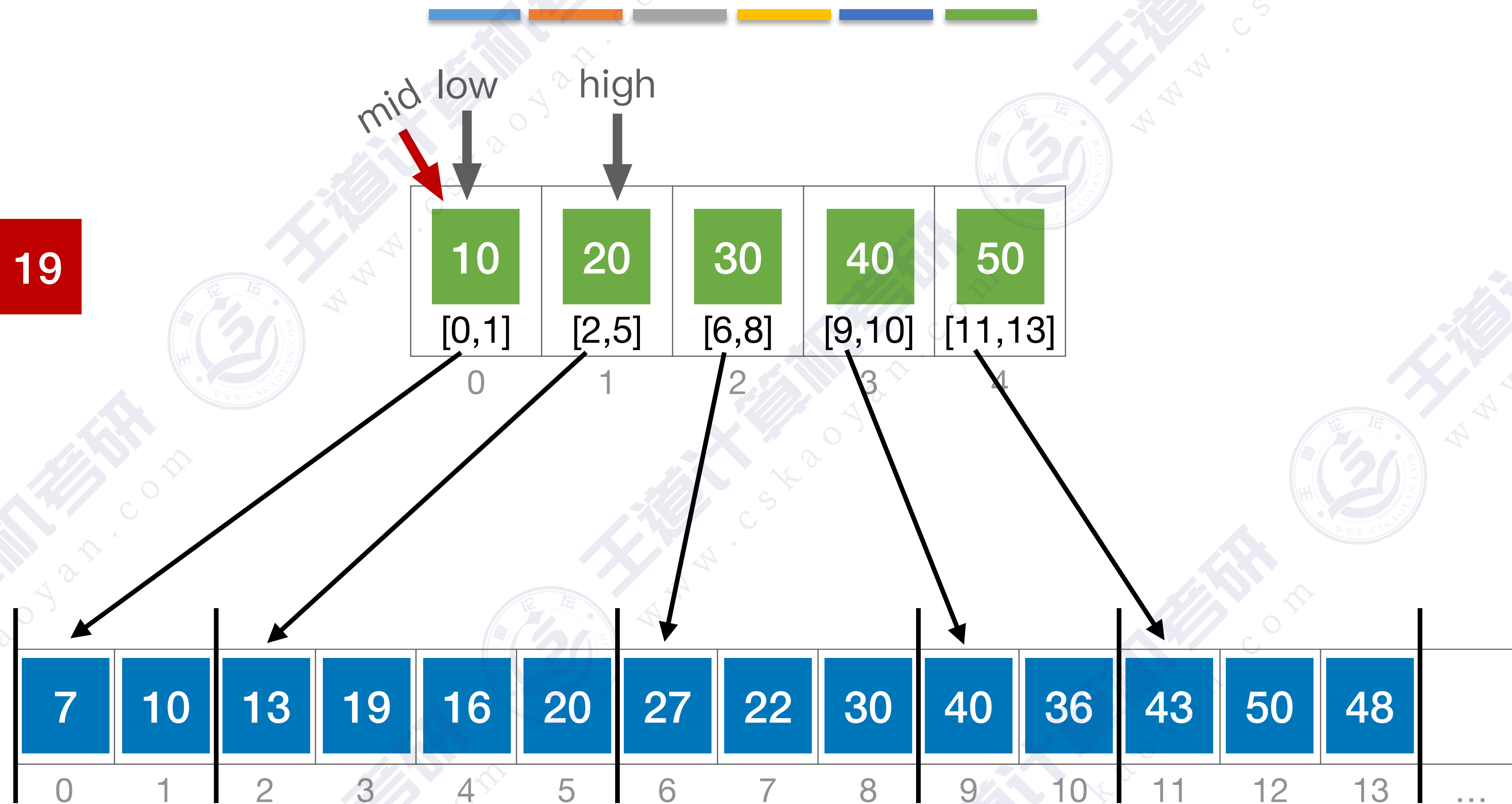
19



用折半查找查索引

查找目标:

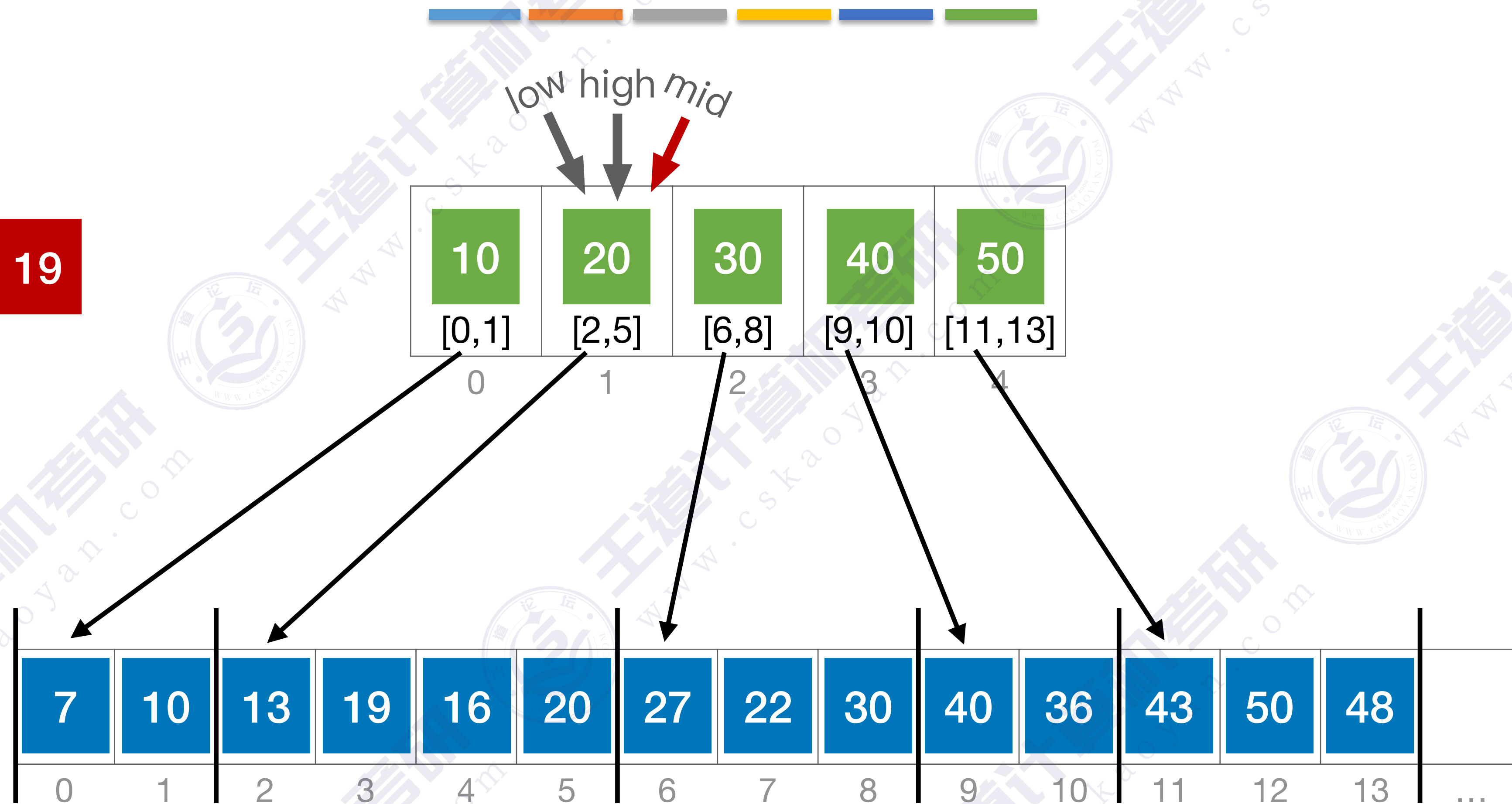
19



用折半查找查索引

查找目标:

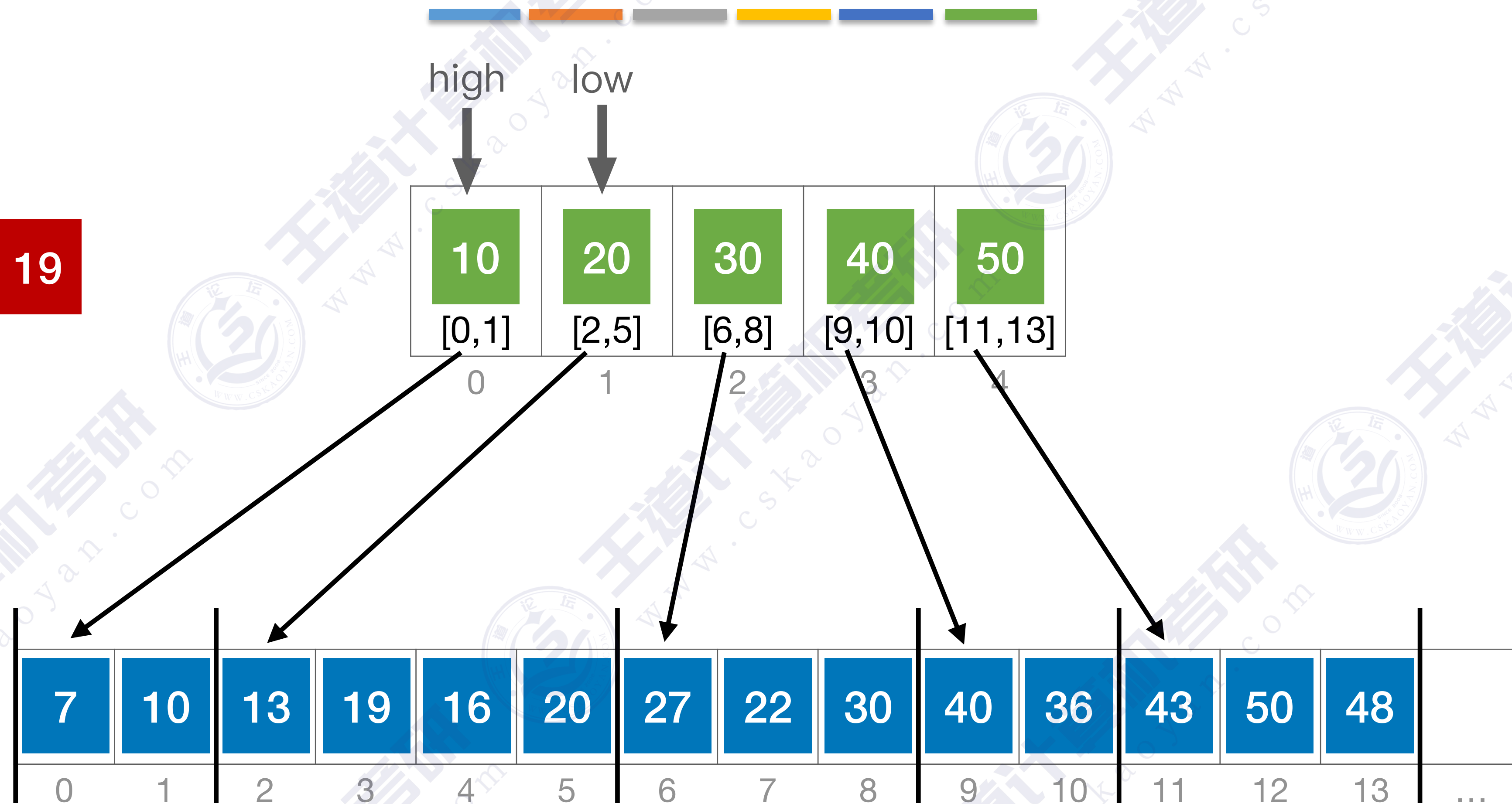
19



用折半查找查索引

查找目标:

19

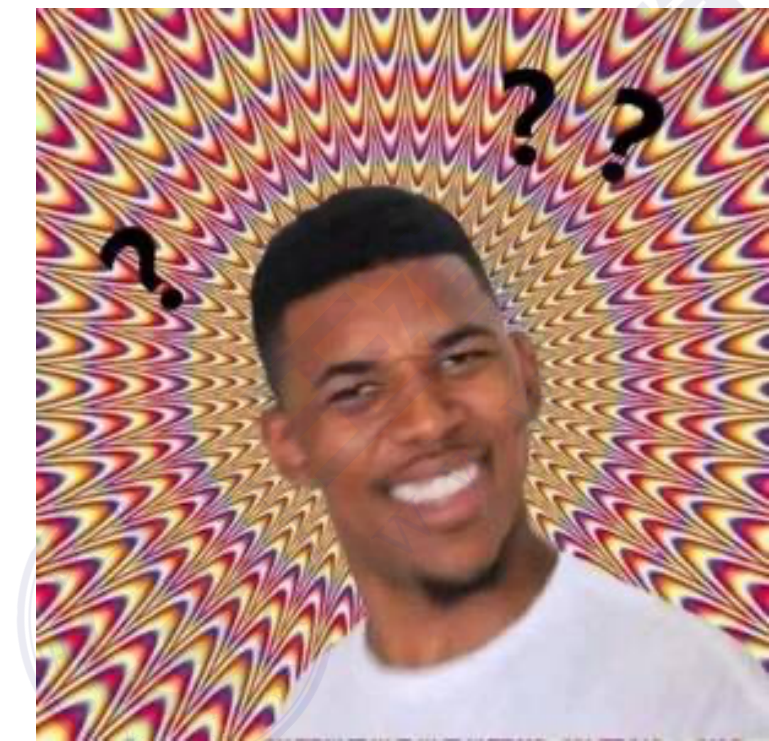
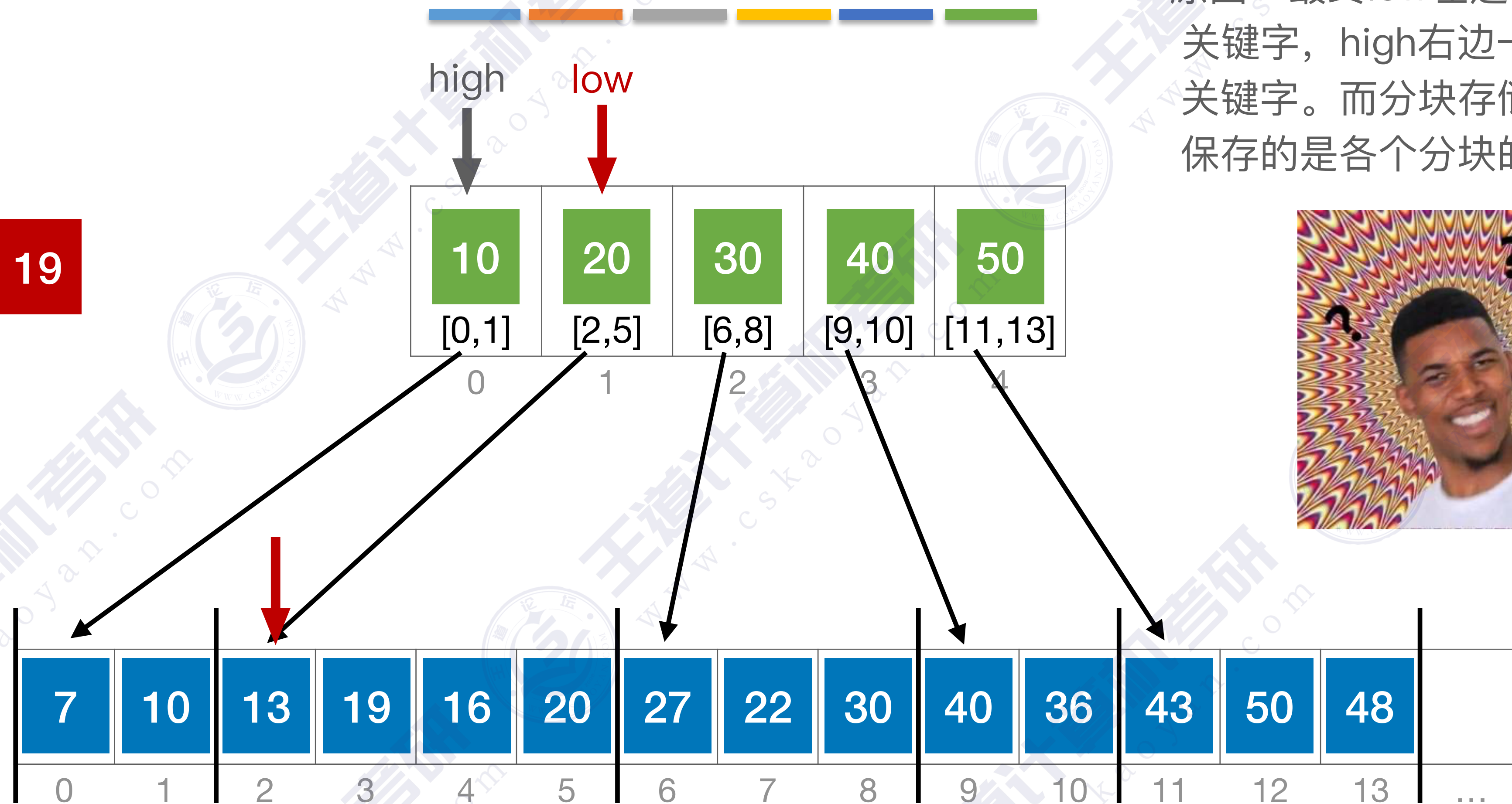


用折半查找索引

原因：最终low左边一定小于目标关键字，high右边一定大于目标关键字。而分块存储的索引表中保存的是各个分块的最大关键字

查找目标：

19

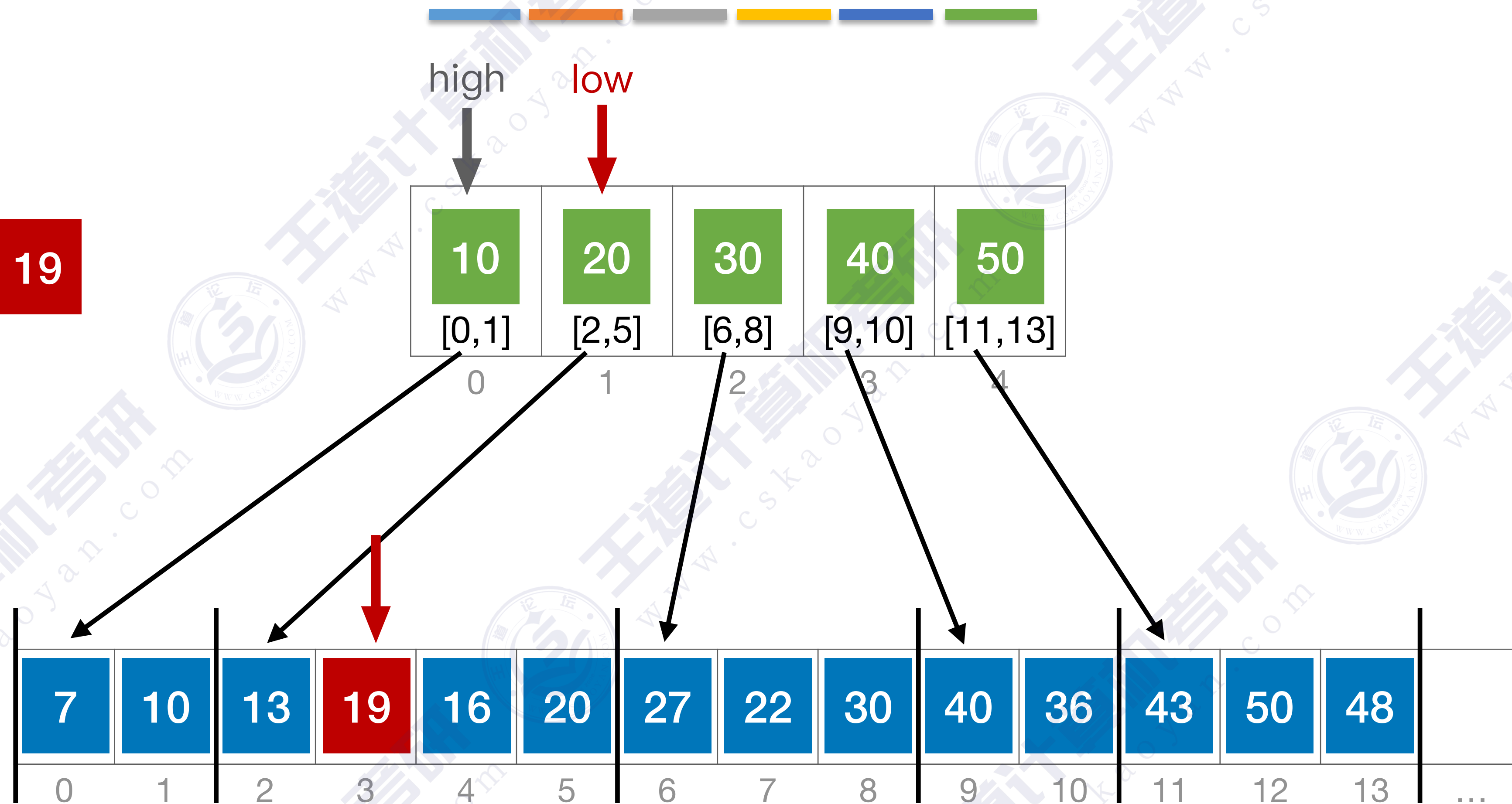


若索引表中不包含目标关键字，则折半查找索引表最终停在 $low > high$ ，要在low所指分块中查找

用折半查找索引

查找目标:

19

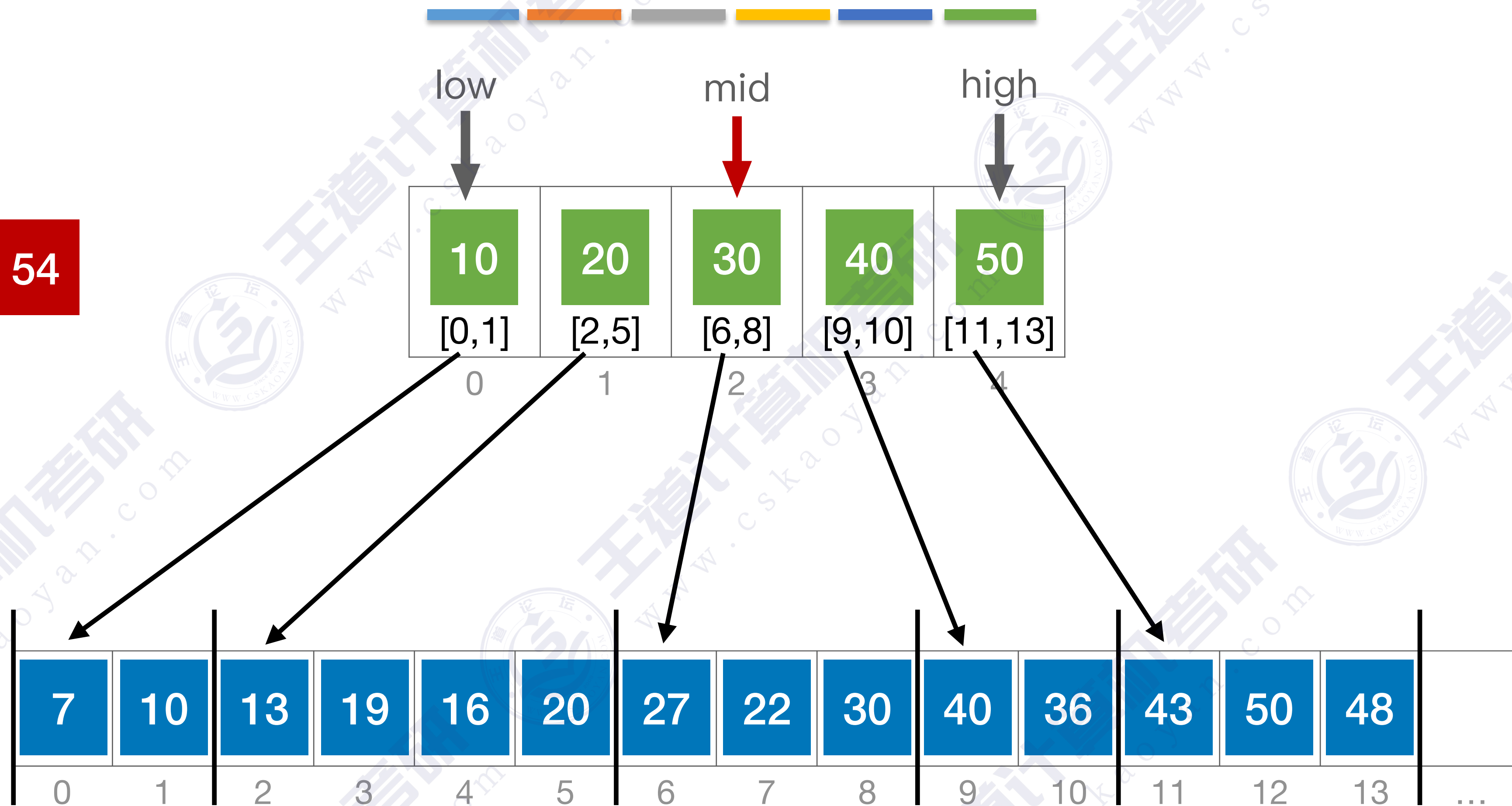


查找成功

用折半查找查索引

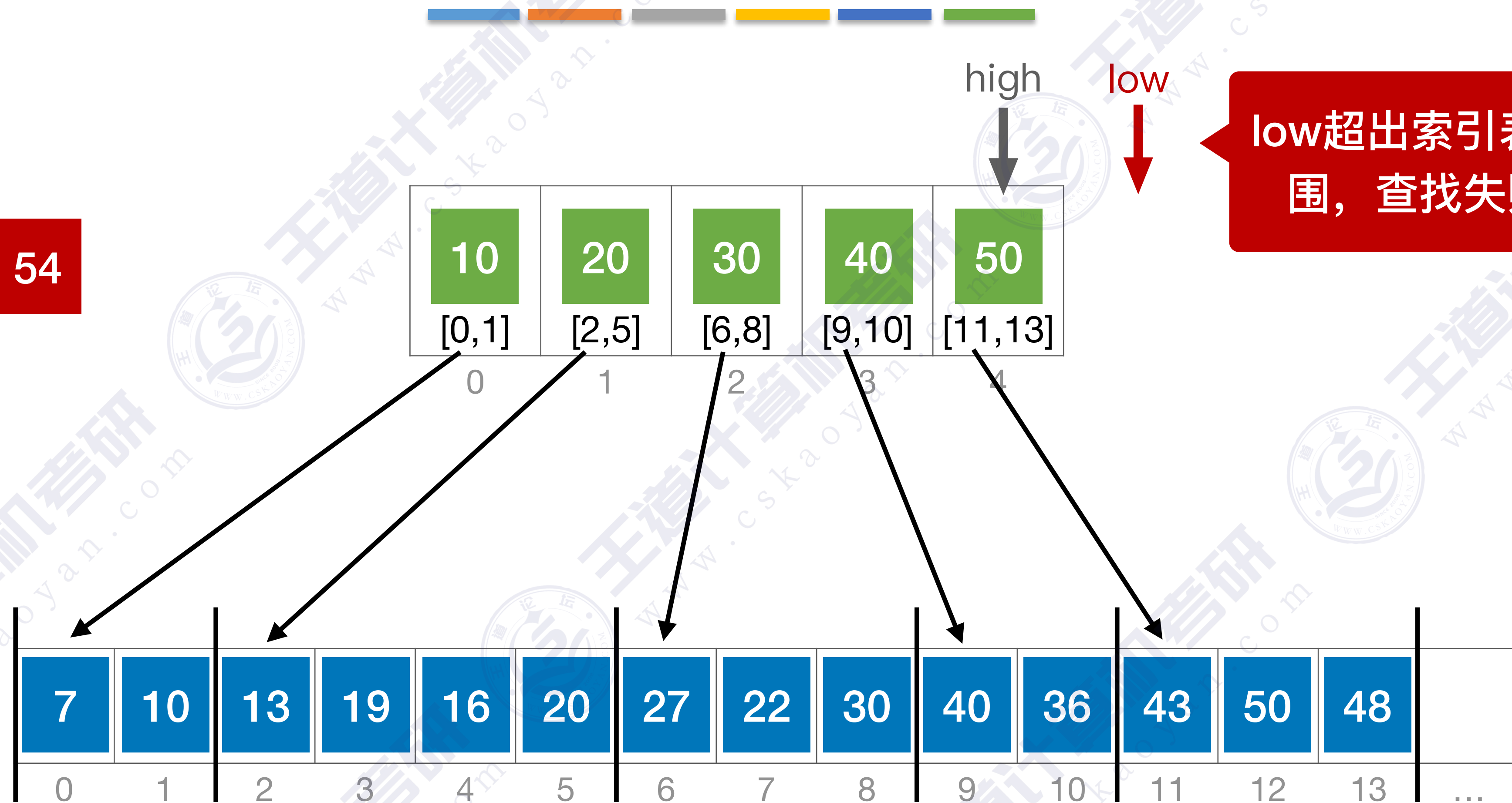
查找目标:

54



用折半查找索引

查找目标: **54**

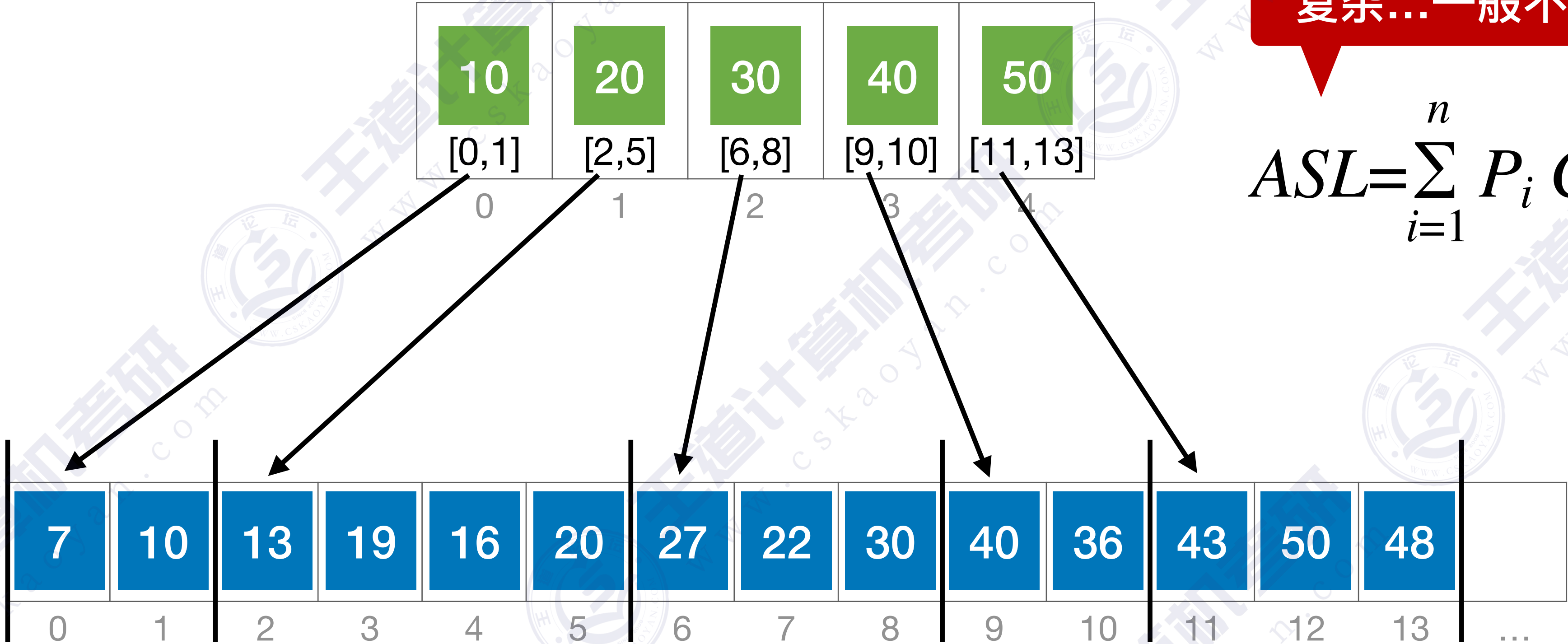


若索引表中不包含目标关键字，则折半查找索引表最终停在 $low > high$ ，要在low所指分块中查找

查找效率分析 (ASL)

查找失败的情况更复杂...一般不考

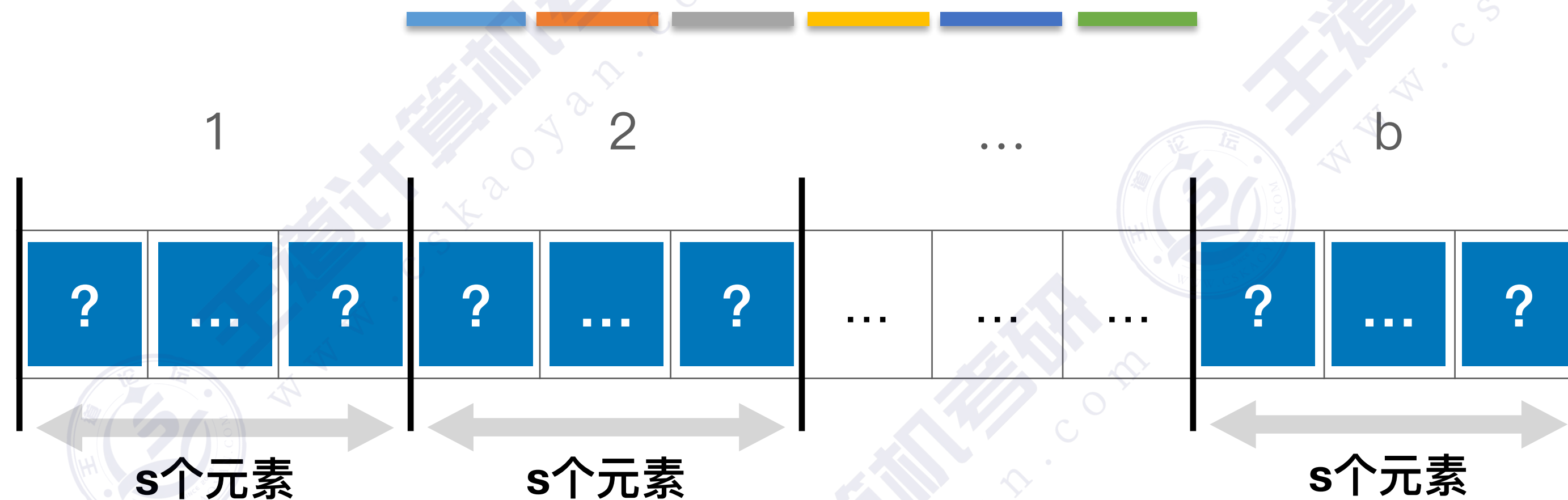
$$ASL = \sum_{i=1}^n P_i C_i$$



共有14个元素，各自被查概率为1/14
若索引表采用顺序查找，则 7：2次、10：3次、13：3次...
若索引表采用折半查找，则30：4次、27：2次？



查找效率分析 (ASL)



假设，长度为 n 的查找表被均匀地分为 b 块，每块 s 个元素

设索引查找和块内查找的平均查找长度分别为 L_I 、 L_S ，则分块查找的平均查找长度为

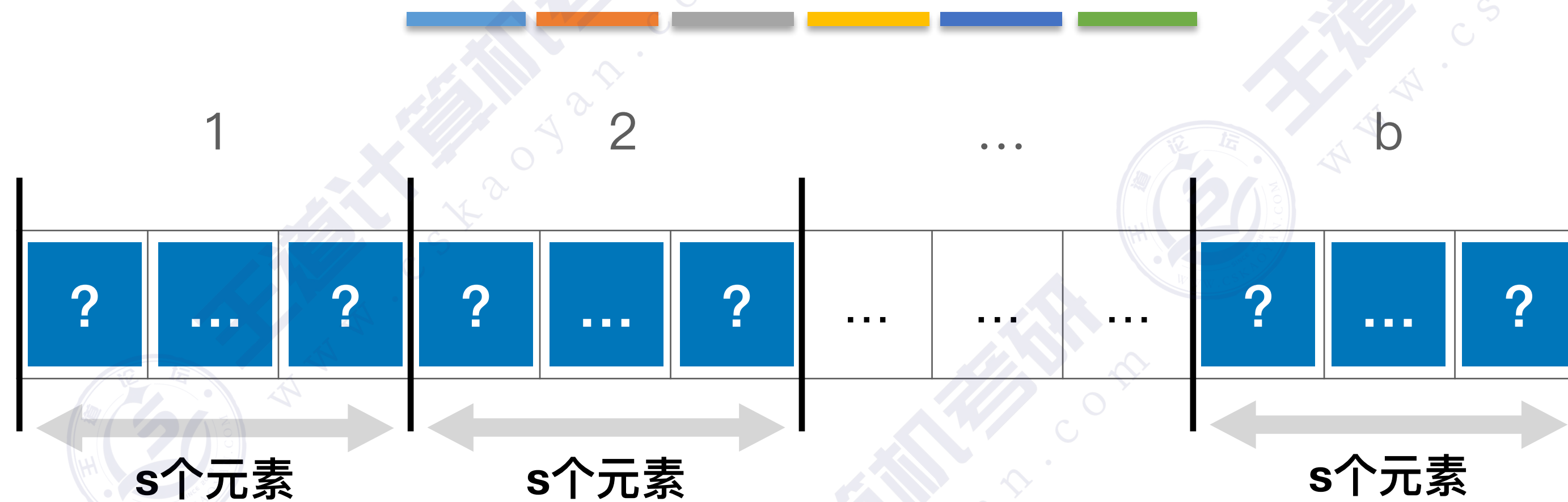
$$ASL = L_I + L_S$$

用顺序查找查索引表，则 $L_I = \frac{(1 + 2 + \dots + b)}{b} = \frac{b + 1}{2}$, $L_S = \frac{(1 + 2 + \dots + s)}{s} = \frac{s + 1}{2}$

$$\text{则 } ASL = \frac{b + 1}{2} + \frac{s + 1}{2} = \frac{s^2 + 2s + n}{2s}, \text{ 当 } s = \sqrt{n} \text{ 时, } ASL_{\text{最小}} = \sqrt{n} + 1$$

若 $n=10000$ ，则
 $ASL_{\text{min}}=101$

查找效率分析 (ASL)



假设，长度为 n 的查找表被均匀地分为 b 块，每块 s 个元素

设索引查找和块内查找的平均查找长度分别为 L_I 、 L_S ，则分块查找的平均查找长度为

$$ASL = L_I + L_S$$

用折半查找查索引表，则 $L_I = \lceil \log_2(b + 1) \rceil$ ， $L_S = \frac{(1 + 2 + \dots + s)}{s} = \frac{s + 1}{2}$

$$\text{则 } ASL = \lceil \log_2(b + 1) \rceil + \frac{s + 1}{2}$$

知识回顾与重要考点

又称“索引顺序查找”，数据分块存储，块内无序、块间有序

算法思想

索引表中记录每个分块的最大关键字、分块的区间

先查索引表（顺序或折半）、再对分块内进行顺序查找

ASL=查索引表的平均查找长度+查分块的平均查找长度

ASL

设n个记录，
均匀分为b块，
每块s个记录

顺序查找索引表

$$ASL = \frac{b+1}{2} + \frac{s+1}{2}$$

当 $s=\sqrt{n}$ 时， $ASL_{\text{最小}}=\sqrt{n}+1$

折半查找索引表

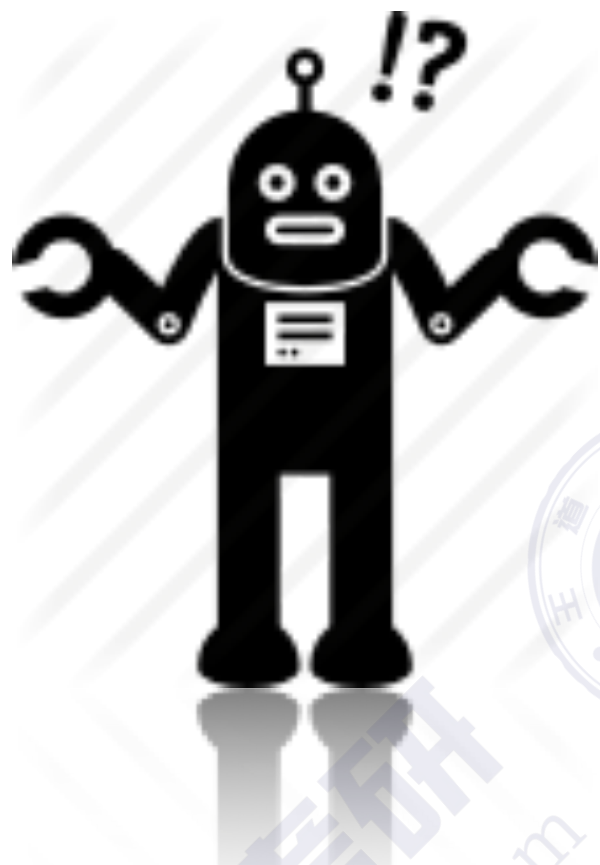
$$ASL = [\log_2(b+1)] + \frac{s+1}{2}$$

易错点

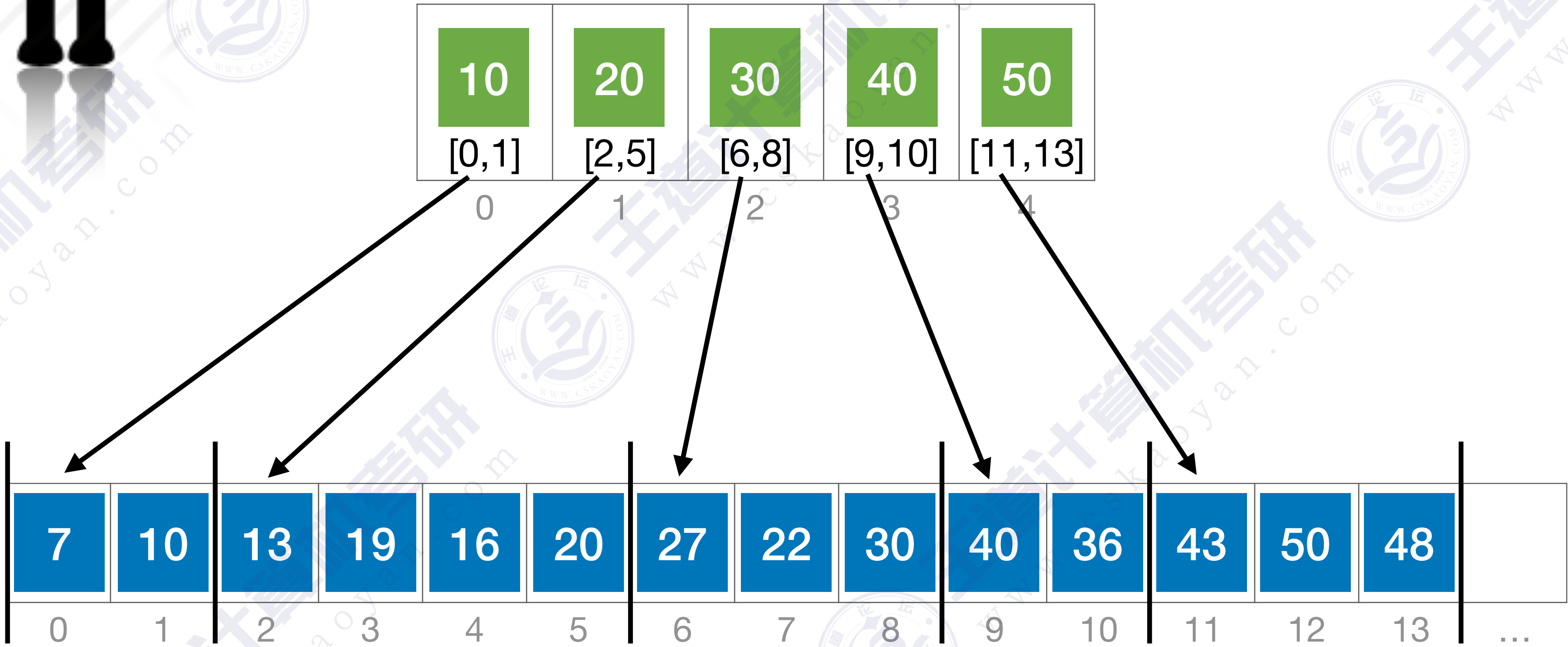
对索引表进行折半查找时，若索引表中不包含目标关键字，
则折半查找最终停在 $low>high$ ，要在 low 所指分块中查找

分块查找

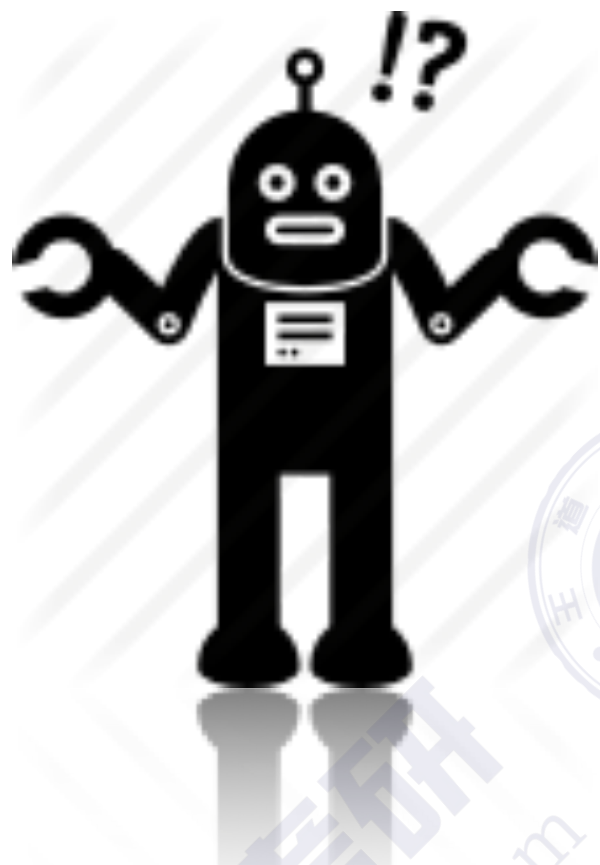
拓展思考



若查找表是“动态查找表”，有木有更好的实现方式？



拓展思考



若查找表是“动态查找表”，有木有更好的实现方式？——链式存储

